Version 3.8.3, October 8 2008

Table of Contents

Overview

What CRUNCH does

CRUNCH analyses electrophysiological and behavioural data. Specifically, it was designed for the analysis of video tracking data, and two types of electrophysiological data: field potentials (EEG) and action potentials (spikes). CRUNCH can be used to combine all three types of data, and to perform some higher level processing, For example, CRUNCH can be used to create place-field plots and to detect theta oscillations in field recordings.

Data analysis can be conducted on any data type (position, spiking, field potentials) independently, but certain analyses require more than one type. For example, place maps require both spike and position data. Spikes are considered the main unit of data, so CRUNCH sets out to assign video-related and field-potential variables to each recorded action potential.

Crunch is a console / command-line driven program - there is no graphical interface, and all output is directed to files beginning with "crunch_" which will appear in the same directory as the data. The advantage to this arrangement is speed and simplicity of use, and the ability to incorporate CRUNCH into batch processing jobs. How CRUNCH behaves is defined in a text file called "Crunch.rc", which must be in the same directory as the executable program file (Crunch.exe).

What CRUNCH cannot do

Crunch does not perform spike detection. That is, if your data is collected as a continuous stream you will need another program to detect and extract the data corresponding to individual action potentials. Also, CRUNCH does not perform higher order data analyses - that is, learning curves, comparison between groups and so on will require other programs. However, because CRUNCH produces plain text output, it is ideal for preparing the data for these types of analyses.

Supported data formats

Axona

.pos - video record
.eeg/.eg2 - continuously sampled record
.1 to .16 - spike capture records
.1.cut to .16.cut - corresponding cluster definitions for spike capture records

NOTE: In order for CRUNCH to produce cell-specific statistics, it is essential that your Axona files be accompanied by an appropriately named “cut-file” (cluster definition file). The cut-file is produced using the “exact-cut” option in TINT, and includes not only the geometric boundaries of the defined clusters, but (importantly) also the cluster to which each spike was assigned in the cluster cutting session. The naming is important, because CRUNCH constructs the cut-file name from the name of the spike files the user passes to it.

Sample Axona files for a hypothetical recording trial 7 which used 4 tetrodes...

tr7.1 tr7.1.cut tr4.pos
tr7.2 tr7.2.cut tr4.eeg
tr7.3 tr7.3.cut tr4.eg2
tr7.4 tr7.4.cut tr4.set

The .set files are not currently used by CRUNCH.

Neuralynx

.Nvt - video record
.Ntt - spike capture record (tetrodes)
.Ncs - multi-channel continuously sampled recording

Neuralynx continuous record files contain records consisting of a timestamp followed by a block of continuous samples - typically 512. The number of samples in each record is contained in the file, and CRUNCH uses this number to “fill in” the timestamps for each sample, assuming an even temporal distribution of samples. The timestamp in each record is taken to be the exact time of the first sample of the block.

Buzsaki/Csicsvari lab

.whl - video record
.res.x - spike capture records - "x" suffix denotes probe number
.clu.x - corresponding cluster definitions for .res files
.eeg - continuously sampled channels (usually field oscillations)

For Buzsaki/Csicsvari lab .eeg files, the sampling rate is assumed to be 1/16th of the broad-spectrum sampling rate - 20KHz/16 = 1250 Hz. There are no timestamps in the .eeg files, just an ASCII list of A/D voltage values. In addition, all channels are multiplexed into a single stream, so it is essential to tell CRUNCH how many channels are present and which channel to analyse using the -echantot and -echannel commands.

While spike times are contained in the .res file, the cluster definitions are contained in a corresponding .clu file. CRUNCH automatically looks for the matching .clu file for each .res file it is given - if the .clu file is missing or has a different prefix, processing will stop and a "file not found" error will result.

Using CRUNCH with Windows

Quickstart

- place a shortcut to CRUNCH.exe on your desktop
- edit the *CRUNCH.rc* file to set -datatype to the type of data you want to analyze
- adjust other settings in the .rc file as required
- drag and drop your data files to the program shortcut

Drag & Drop Mode

Select the files you want to analyse using the left mouse-button while holding the down the [CONTROL] key. Then drag the files to a shortcut to CRUNCH.exe. In Windows, if you select files and drag them to the CRUNCH icon, the command line generated contains the full name of the program (including the full path to the folder containing it), and the full path to each of the data files you selected. CRUNCH looks to the program folder for the CRUNCH.rc file, and sets the folder containing the data files as the destination for output. You cannot override the CRUNCH.rc file defaults in drag & drop mode. Therefore, to change how CRUNCH works, you must alter and re-save the CRUNCH.rc file, using any basic text editor, such as Notepad.

Batch files

Windows batch files can be used to automate large amounts of data processing, or to simply repeated analysis of the same data while varying parameters in the CRUNCH.rc file. Windows batch files also allow you to override the CRUNCH.rc file settings by manually defining a command line.

Here is an example Windows batch file for CRUNCH....

```
@ECHO off
set USERSETS= -vidres 5.87 -setverb 1 -maptype 1 -mapsmooth 0 -mapbinsize 2.5 -setfilter 1 -pause 0 -clean 0
set datapath=D:\Work\Kainate\Original_Data\hux081-day07-UBP304\hux081-day07-tr1\

D:\Work\Bin\Crunch\Debug\Crunch.exe  %pd%VT1.Nvt %pd%CSC4.Ncs %pd%R_Sc1.Ntt %pd%R_Sc2.Ntt %pd%R_Sc3.Ntt %pd%R_Sc4.Ntt %USERSETS% -suffix "081 day07 UBP304 1"

type %pd%crunch_cells.txt > D:\Work\Kainate\crunch_cellsummary.txt

set pd=D:\Work\Kainate\Original_Data\hux081-day07-UBP304\hux081-day07-tr3\

D:\Work\Bin\Crunch\Debug\Crunch.exe %pd%VT1.Nvt %pd%CSC4.Ncs %pd%R_Sc1.Ntt %pd%R_Sc2.Ntt %pd%R_Sc3.Ntt %pd%R_Sc4.Ntt %USERSETS% -suffix "081 day07 UBP304 2"

type %pd%crunch_cells.txt >> D:\Work\Kainate\crunch_cellsummary.txt
```

In this example, the variable “pd” is used to simply writing the directory in which the data is found. There is no need to execute a command to change to this directory, because the full path length is specified for each filename using %pd% as a suffix.

Using CRUNCH with Linux/Unix

Quickstart

- edit the *CRUNCH.rc* file to set -datatype to the type of data you want to analyze
- adjust other settings in the .rc file as required
- in a terminal window, navigate to the directory where your data is
- type the full path to *CRUNCH.exe*, followed by the names of the files to analyze
- eg.

```
/usr/local/bin/crunch/CRUNCH.exe  videofile.nvt spikefile.ntt eegfile.ncs
```

Command line

Type the full path to CRUNCH at the command line, followed by the names of the files you want to analyse, and then the command line overrides. Usually in Linux or Unix, the name of the program itself is sufficient because the $PATH environmental variable is configured to tell computer where to look for programs. However, CRUNCH requires the full path in order to find the CRUNCH.rc file. One solution is to create a simple batch file which redirects to CRUNCH.exe. (click here for details).

You can type the full path to the data files you want to analyse on the command line, but this is unnecessary if you first navigate to the directory containing the files. CRUNCH assumes that the first arguments following the program name are file names, up to the first argument beginning with a hyphen (-). The file names themselves can occur in any order, and the file name extension (the series of letters following the first dot in the filename) is used to determine the file type. Wildcards may be used.

Batch File

Batch files can be used in Unix/Linux to automate analysis of large amounts of data, but also to allow different users to make shortcuts with their own CRUNCH settings. Batch files created in a directory identified by the $PATH variable can then be used to execute CRUNCH without typing the full path to CRUNCH on the command line.

How CRUNCH Processes Data

Input Files and Control Variable Arguments

Whenever you run a program, on any computer, an execution command-line is generated. The command line consists of a number of elements, or "arguments". The first argument is the name of the program itself. Additional arguments include the name of files the program must open, or commands which determine how the program behaves.

However CRUNCH is used, and on whichever operating system, ultimately a command line is created which looks something like this example...

/drive/directory/CRUNCH.exe	file1.Nvt file2.Ncs file3.Ntt	-maptype 0 -verbout 1 -vidres 3.115
full path to the crunch program	names of files to be analysed	variable name/value pairs

Note that the first argument is always the full path of the program name. CRUNCH expects the next command line arguments to be the names of files to open, followed by commands which can override the contents of the CRUNCH.rc file. If there are no additional arguments, a summary of CRUNCH usage is printed to screen.

IT IS IMPORTANT TO REMEMBER THAT FILENAMES (arguments not beginning hyphen) MUST COME BEFORE ANY ADDITIONAL ARGUMENTS (those beginning with a hyphen).

The *CRUNCH.rc* resource file

CRUNCH expects the first command line argument to specify the full path (including directories) to the program. This information is used to find the *CRUNCH.rc* resource file, which sets the default behaviour of the program. If CRUNCH cannot find this file in the program directory or folder, it will abort.

The resource file contains a list of variable-value pairs which represent the full range of controls available to the user. Each of these controls can be overridden on the command line, using exactly the same syntax as appears in the resource file. CRUNCH expects the first word (block of non-whitespace characters) on each line to be a variable name (beginning with a hyphen), and the second word to be the value you want to assign to that variable. Lines beginning with # are treated as comments to be ignored, as are additional words on each line.

Time-stamping

For all data types, representations of time (usually timestamps accompanying each record) are converted into real times in seconds. CRUNCH uses double-precision floating-point numbers to store times. This requires more memory space, but considerably simplifies subsequent calculations of speed, acceleration, etc. However, synchronisation of timestamps between the position, EEG and spike records is presumed to be done by the data acquisition software, and differs depending on the data type as follows.

Neuralynx Files

Each binary data type is accompanied by a timestamp from a common clock. The time is related to the time since startup on the acquisition computer, and is not zero-aligned at the start of each recording session - that is, the time of the first record in each data file need not be zero, and usually isn't. EEG records include a single timestamp for each block of (typically 512) EEG samples. CRUNCH generates intermediate (interpolated) timestamps for each of the 512 individual samples.

Csicsvari Files

All times are expressed as time relative to the first sample in the continuous recording (time zero). EEG files contain no timestamps, but are resampled from the original continuous recording at 1/16th the original rate. Position data is resampled and padded to create one record for every 512 continuous recording records. Therefore, CRUNCH generates timestamps for these data typpes based on the assumption of regular sample intervals. Spike files are the only records with accompanying times - these are relative to the first continuous record (time zero).

Axona Files

CRUNCH generates position and EEG timestamps based on the assumption of a constant interval between samples, the first sample of each type corresponding to zero. Spike times are taken from the binary data files and are taken as relative to the first EEG sample (time zero).

Video (Position) Data

There are several steps to processing video data, which always occurs in the following sequence:

1. Detection of the tracking LED
2. Conversion of pixel values to centimeters and y-inversion of the dat
3. Optional shifting of the data along the subject's axis of motion or head-direction
4. Invalidation of "jumpy" position samples, usually caused by LED reflections or mistracking
5. Optional interpolation of data across invalid tracking points
6. Calculation of velocity and
7. Creation of a smoothed copy of the position record for calculating heading, angular velocity, etc.

These steps are discussed in greater detail below.

Detection of the tracking LED

This is done differently for different data formats. For Neuralynx data the user can specify the number of LED's to be detected (up to 5) with the -nleds variable. These "targets" are then used to assign a value to a red, green and blue spot. The user can define which spot best represents the animal's position with the -pled variable (1=red, 2=green, 3=blue). If colours are not defined in the file the user may chose -pled 0, which sets the largest detected spot as the animal's position.

For Axona and Csicsvari video files an extracted x/y position is specified in the file, and this is the value CRUNCH uses.

Video resolution conversion

Position data is converted from x,y camera pixels to centimetres using -vidres, which sets the number of pixels per cm. If this variable is not set correctly, all calculations of speed, field size, etc. will be inaccurate, as will several behavioural filter settings. Whenever tracking is lost, both x and y coordinates are set to -1.

Inversion of camera coordinates

Most video cameras use a coordinate system in which 0,0 corresponds with the top left of the camera frame. However, Cartesian coordinate systems and the mapping space of most graphic formats (including postscript) define 0,0 as the lower left corner. For this reason, CRUNCH inverts the camera y-coordinates at the time when the data is first read into memory. For all calculations, 0,0 is the bottom left hand corner, and this is reflected in the postscript files CRUNCH produces.

Position shifting

Data can be shifted along the trajectory of the subject using -pshiftt, or along the axis of head orientation using -pshiftd if head direction information is available. This is to correct for potential mis-placement of the

tracking LED's on the subject's head. For example, if the LED's are too far back, the subject's position should be shifted forward.

Jumpy tracking points

Sometimes reflections and light pollution will cause the target to appear to jump from one position to another. CRUNCH handles jumpy data by imposing a maximum velocity on position data, via the -dejump variable. If the distance between two successive points is too great for the time interval between them, it is unreasonable to expect that the target could move at that speed. Tracking data will be set to -1 if the speed threshold is exceeded.

Position interpolation

CRUNCH can interpolation across gaps in the position record occupied by "invalid" records, filling them with points lying on a straight line between the previous and next valid position sample. The -pinterp variable determines the maximum period of lost tracking across which position values will be interpolated. The default is 0.4 seconds, which is also the length of time over which running speeds are integrated. Interpolation is performed after de-jumping, which helps to reduce the chances of interpolation producing artefactual trajectories.

Position smoothing

Smoothed versions of the position record are not output by CRUNCH, but are used in the calculation of some variables like running speed, angular position, and angular velocity. The method uses a sliding window, the size of which is controlled by the -psmooth variable. Smoothing replaces the original value at a given time with the average of the values from the surrounding times. This prevents high-speed fluctuations in these two variables due to minor head movements or camera image jitter.

Calculation of path lengths

Path lengths are calculated using a "cord" method for integration across time. Integration is necessary because path lengths are theoretically infinite if you sum distances across small enough intervals - if you are interested, read up on the "**Length of the Coastline**" problem!

Values are derived from the interpolated but unsmoothed position data - however the -psmooth variable sets the size of the window across which motion is integrated. For example, if -psmooth is set to 0.4, then the position record is cut into non-overlapping 0.4 second chunks. In each chunk the distance between the first and the last position sample is calculated, and that distance is divided eaqually amongst each position sample in the chuck - that is, for each position sample there is a "distance travelled since last sample" stored in memory. If the start or end samples in a given chunk are invalid tracking points, then no distance is calculated. While path lengths can be calculated even if there is an invalid sample in the middle, in practice this scenario never arises provided -pinterp is set to the same value as -psmooth.

Calculation of velocity

Velocity is also integrated across a cord of length (duration) defined by the -psmooth variable. However in this case a sliding cord (window) is used, and the velocity assigned to each position sample reflects the length of cord centred on that sample divided by -psmooth. Velocity like path lengths, must be integrated so that pixel-jitter in the camera signal and brief artefactual or "wobble" movments can be ignored.

A note on multi-spot tracking

The Neuralynx system tracks up to 50 targets (0-49, in decreasing size order). CRUNCH considers only the two largest targets, and only those which contain red (but not blue) pixels or blue (but not red) pixels. The red target is taken as the position of the animal. The blue target, if present, is used to calculate LED distance. LED distance is set to “-1” if either the red or blue target is untracked. At present, the second LED is not used to calculate head direction.

Field Potential (EEG) Data

Continuously acquired (EEG) files typically do not have a timestamp associated with every sample. Therefore, timestamps are interpolated between those present in the file. A fixed rate of acquisition is assumed. Note that EEG data can be scaled using the -escale option, which can also be used to correct inverted EEG value, if a negative number is used.

Cycle detection

[coming soon]

Action Potential (Spike) Data

Inter-Spike Interval (ISI)

For each spike, the ISI is calculated as the time between that spike and the previous spike from the same cluster. This is an inherently noisy (highly variable) measure, but is essential to burst-analysis.

Spike-in-burst and cell burstiness

A 2-6 ms ISI between any two spikes from a given cell is taken be indicative of burst firing (Ranck, 1973; Harris & Hirase, 2001). The -setburst varbiable can be used to specify the maximum interspike interval tolerated within a burst. A number is assigned to each spike to indicate its number within a burst, with zero indicating that the spike is not part of a burst. Cell burstiness is calculated as the proportion of cell spiking which is bursty (Ranck, 1973). Burstiness is calculated only for data passing behavioural filtering.

Instantaneous Firing Rate (IFR)

Instantaneous firing rate (IFR) is a more reliable measure of the activity of a cell at any given moment in time. It is the boxcar-averaged ISI within a 250 ms time window. Although CRUNCH deliberately avoids tying this measure to EEG oscillations (which can be variable or indeed absent), the window size corresponds with approximately 2 theta cycles, assuming an 8 Hz oscillation.

Spike Behavioural Attributes

CRUNCH assigns position, speed, movement direction, head direction and angular velocity values to every spike. These are taken directly from the position records. Because the spike sampling rate is significantly higher than the position sampling rate, the position for any given spike is taken as the interpolated position based on the nearest video samples on either side of spike, and the time elapsed in the video inter-sample interval. This provides highly accurate estimation of the positions at which spikes occur.

However it also means that, rarely, spikes can be attributed to positions the animal never actually visited. This requires a peculiar movement trajectory around a region of the maze the rat never visited combined with spiking from the cell in question, but it may account for some loss of spikes, not from the main spikes record, but from the place map data.

Place field analysis

Generating firing rate maps

Position data (cm) is first converted into map-bins, with the size of each bin determined by the -mapbinsize variable. A dwell-time is then calculated for each bin in the map, but bins with less than -mapmindwell position samples are treated as being unvisited. This helps to avoid spuriously high firing rates in bins where sampling was limited to (say) a single position sample.

Next, a firing rate map is generated for each cell by dividing the number of spikes which occurred in each bin by the total cumulative dwell-time in that bin. Unvisited bins are assigned a rate value of -1. Once the rate array is generated, it is used to calculate the information content, sparsity and coherence of the spatial

firing pattern.

Next the firing rate map is smoothed to compensate for sub-optimal position sampling. Smoothing is accomplished using a boxcar averaging algorithm. From the smoothed map, the peak firing rate is determined. With a reasonable bin size (eg. 2.5 cm) and smoothing factor (eg. 2) this yields a reliable estimate of the peak firing rate.

Place field definition

The place field for a given cell is defined as the region of contiguous bins adjacent to the peak firing rate bin which all have a firing rate greater than a percentage of the peak rate. That percentage is defined by the -mapthresh variable. 10% is recommended. The peak zone of the place field is also determined using a threshold percentage of the peak rate, as defined by the -mappeakzone variable. 75 % is a typical value.

Place fields (and runs through them) are defined using the smoothed rate map, and the peak bin is taken as the bin with the highest firing rate. Fields should be defined using smoothed data to avoid under-detection of infield bins due to variability in the rate of neighbouring bins. Smoothing will assign rate values to unvisited bins. Therefore it is also advantageous to use a smoothed rate value to define the peak (and hence the threshold for including bins in the place field).

Behavioural Filtering

Filtering is a way of restricting analysis to particular portions of the data set. If the -set_filter variable is set to “1”, filtering will automatically be applied to all periods when there is no video tracking (usually because the tracking LED is obscured). In addition, other parameters can be defined as follows....

- start time (seconds)
- end time (seconds)
- minimum velocity (cm/s)
- maximum velocity (cm/s)
- heading (preferred heading +- tolerance value)
- angular motion (clockwise, counterclockwise)

Effect on position data

The following variables are invalidated...

- velocity
- direction

An invisible variable (pos_filter) is set from “0” to “1”. There is no effect on the “x” and “y” positions.

Effect on EEG data

Filtering occurs after oscillation smoothing and cycle-fitting, so these functions are not affected by momentary changes in behaviour. However after fitting, the following variables are set to “invalid”...

- EEG voltage
- EEG phase

Therefore, estimates of oscillation shape and oscillation frequency do not include data from times when the filter criteria are not met.

Cycle start- and end-times output to the file *crunch_eegcycles.txt* include all detected cycles, but cycle records are tagged with a "0" if they pass the filter criteria or "1" if the criteria are violated at any time during the cycle.

Effect on spike data

Spikes which occur during periods failing the behavioural criteria are designated to cluster zero, and are therefore excluded from all subsequent cluster-specific calculations, including spike counts, place field maps, etc. In practice, this means that if the position sample ***just prior to*** a spike fails the filter criteria, then that spike is set to cluster zero. **NOTE** however that spikes are assigned interpolated versions of the positional data (x, y, velocity, direction etc.). As a result, a small proportion of spikes occurring near the start of a filter violation may be assigned positional data values which appear to violate the filter settings. This is normal and acceptable.

Effect on place maps

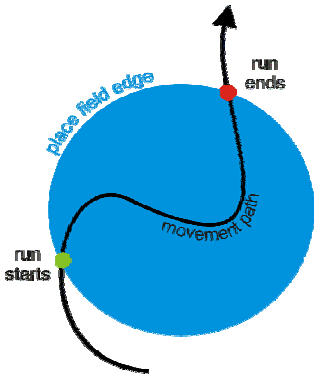
Spikes occurring during behavioural epochs failing the behavioural filter are reassigned to cluster zero, and do not contribute to place maps. However, the dwell-times for the map are also adjusted accordingly - that is, time during "failed" behaviour is not included in the dwell-time map for calculating firing rates. If this were not the case, actual firing rate values in the rate map would be inaccurate.

Effect on runs (see below)

Crunch does not filter runs through a cell's place fields, except by excluding runs in which tracking was interrupted. Therefore, all entries and exits are included in the crunch_runs.txt output. However, spikes occurring during filter violations will have already been removed.

As a result, it is best not apply filters for specific speed ranges, headings and so on if the effect of these variables on spiking during runs is to be analysed. Instead, use the average values for speed, direction etc. as found in the crunch_runs.txt output. These values can be used to select runs meeting behavioural criteria with the RunsFilter.exe utility.

Runs-Analysis



Definition of a run

Runs-analysis creates a set of time windows in which the subject is traversing the place field of a given cell. A run begins at the time of the first position sample after the subject crosses the field boundary (point "**A**") and ends when the subject exits the field (point "**B**"). Runs will be rejected if they include untracked positions. Runs analysis will only be performed if the variable -output_runs is set to "1", in which case a "crunch_runs.txt" output file will be produced.

Like the place fields themselves, runs are defined using the smoothed version of the firing rate map. This means that data may be included for map bins which the rat never actually visited, and it reduces the probability that runs will terminate mid-place-field if such a bin is crossed. However, the *crunch_matrix.txt* output file may have valid firing rates for visited bins if the -mapfill option is set to "0". If you are using the matrix output to do further analysis of runs, this needs to be kept in mind.

NOTE: a run can only be terminated (and a new one begin) if the subject leaves the place field. This can cause problems for place fields near the edge of the environment on linear track or radial arm maze tasks. For example, if in post-processing run-inclusion depends on direction or minimum-speed criteria, very few runs through fields at the end of an arm will be included.

Run characteristics

In addition to being identified with a particular probe, cell, start time and end time, each run is characterised in the following ways:

Run duration is the time elapsed between field entry and field exit.

Run length is the sum of the lengths of the segments connecting the position samples A and B (see “Definition of a run”). **Run velocity** is the mean speed between A and B.

Run avel is the mean angular velocity during the run, positive values representing counter-clockwise runs, and negative values clockwise runs.

Run direction is the circular mean heading of the subject during the run.

Run dirsd is the circular standard deviation of direction during the run. Values of standard deviation of 45 or less correspond with increasingly “straight” runs, while values above 45 (typically up to 90) correspond with increasingly tortuous runs.

Run spikes is the number of action potentials fired by the cell during the run.

Run rate is simply (run spikes) / (run duration).

Run centre designates whether a particular run passed through the centre of the place field. "Centre" means any bin in the top firing rate percentile inside the field, the threshold percentile defined by the -mappeakzone variable.

Run maxbin designates the highest firing rate bin travelled through on the run, expressed as a prportion of the peak firing rate. This can serve as an alternative way of determining whether a run passes through a high-rate portion of a place field.

Run stop indicates whether (1) or not (0) a run includes periods when the subject’s speed dropped below the minimum velocity specified by -f_velmin. Note that this criterion only applies if -set_filter is also set to “1”.

Spike run-variables

Several new spike variables are calculated if runs analysis is performed. Each spike is assigned the following variables associated with runs through their place field:

Run number (starting with run zero, -1 = spike outside of runs).

Time-in-run (seconds).

Distance-in-run (cm). Spike run variables are output to the crunch_spikes.txt output file so that they may be used for further analysis using other programs. **NOTE:** the crunch_spikes.txt file may appear to be missing data from some runs for a given cell. This will happen if no spikes were fired on that run or if the spikes were removed by the behavioural filter.

Appendix A: CRUNCH Controls

Users can define control variables in two ways:
1) in the *CRUNCH.rc* resource file, which is found in the same directory as the executable (Crunch.exe)
2) on the command line if using a test terminal or a batch file to execute the program.

NOTE: altering the CRUNCH.rc file will change the way the program works for all users
NOTE: variables defined on the command line will override the settings in the *CRUNCH.rc* file.

When processing datasets with different camera settings, environment sizes, etc., it is recommended to create a batch file in which the appropriate settings for each data set is specified on the command line.

General CRUNCH behaviour

-datatype [1|2|3]: The single most critical variable! This tells CRUNCH the format of the data you are working with. This variable is used to check for the appropriate file-extension on the data files sent to CRUNCH, and to make sure those files fit the appropriate data type format.
1: Neuralynx files (default)
2: Csicsvari lab files
3: Axona (DACQ) files

-clean [0|1]: determines whether CRUNCH removes the intermediate postscript files used to generate the summary report. This helps keep your data directories free of clutter. The default is 1 (YES), but set to 0 (NO) if you want to use the individual place field and histogram plots for each cell.
0: no
1: yes (default)

-pause [0|1]: Wait for user to press [RETURN] on completion. In Linux/UNIX it may be convenient to turn off the pause option. However, in Windows setting this option to “no” will cause the output screen will disappear immediately when CRUNCH finishes processing. Note also that if the option is set to “yes”, it will prevent batch processing using CRUNCH, as the program will pause after each invocation.
0: No
1: Yes (default)

-setverb [0|1|2]: sets level of verbose output.
0: Minimal screen output. Summary file *crunch_verbout.txt* written to program directory.
1: Minimal screen output. Summary file *crunch_verbout.txt* written to data directory.
2: Detailed screen output, no summary file produced (default)

Position processing

-cx [-1|position (cm)]: Define the x-position (cm) of the centre of the behavioural apparatus. This value is used for calculation of angular position in polar coordinates. If this option is set to -1, the program will use the mean x-position as the centre.
-1: auto (default)
0-999: cm

-cy [-1|position (cm)] : As above, but for the y-position of the centre. Default: -1.

-dejump [-1|speed (cm/s)] : invalidates position samples which are too far from previous sample given the time elapsed - as expressed by a maximum speed in cm/s. During a “jump” that persists for a long time, eventually sufficient time elapses so that the “jumpy” movement can be treated as valid again. Set dejump to -1 to disable this function

-nox [0|1]: Collapse data in the x-dimension. If -nox is set to 1, all x-values are reduced to values from zero to one. This modification occurs **after** determination of angular position, velocity, movement direction and so on, and does not affect filtering based on these parameters. It is applied **before** assignment of position values to spikes, and therefore will alter the definition and appearance of place fields. This can be useful for linear environments when data sampling is sparse or when there are outliers in the other dimension - in these instances collapsing the data to the "relevant" dimension improves the reliability of firing rate analyses. In such linear environments the user might also consider using the
0: no (default)
1: yes

-noy [0|1]: As above, but for y-values

-pinterp [-1|time(s)] : Apply linear interpolation across invalid position samples, provided the duration of the gap is less than "time". This step follows de-jumping (see above). We recommend using the same value for -pinterp as for -psmooth (below). Set to "-1" to disable.

-nleds [Neuralynx files only] determines the number of distinct LED targets to be considered. If for example -nleds is set to "3", CRUNCH will only analyse the three largest spots detected in any given video record.

-pled [Neuralynx files only] determines which of the tracked LED's best represents the subject's position. If colour data is saved in the Neuralynx .Nvt file, the user can specify red, green or blue by setting -pled to 1,2 or 3 respectively. **NOTE**: CRUNCH will only use spots which are not contaminated with other colours, so set the tracking camera, ambient lighting and the recording system carefully beforehand to get the best colour fidelity. Not that if more than one spot of a pure colour is detected, the largest of these is the one CRUNCH will use. If colour tracking is poor, or colour information is not contained in the file, set -nled to zero - then CRUNCH will always use the largest detected spot for the subject's position.

-posfreq [sampling frequency (Hz)]: (Csicsvari files only) defines the sampling frequency for position records. This is essential for position records which are simply a srteam of x-y coordinates, where the sample number is used to reconstruct a timestamp to correspond with each position record.

-psmooth [time (s)]: this value is applied to position data for several calculations:
1) the size of the sliding window used to integrate position data for the calculation of velocity
2) the size of the non-overlapping chunks of position data used to calculate path-lengths
3) the size of the sliding boxcar averaging window used to calculate movement direction and angular velocity
The default (recommended) value for -psmooth is 0.4 seconds. Note that this variable has no effect on the actual the position values themselves, or the position output sent to the *crunch_pos.txt* file.

-pshiftd [cm]: position-shift along head-direction. This corrects for misplacement of tracking LED's by projecting animal's position along its head-direction by a number of centimetres. Use positive values to shift position forward, negative to shift backwards, zero to leave position values unaltered. **NOTE**: This will affect angular position as well, but not other position variables like velocity, heading and angular velocity.

-pshiftt [cm]: position-shift along trajectory. This is an alternative to -pshiftd, for cases where head-direction information is not available. Based on movement trajectory, this shifts positions to future or past locations by distance, irrespective of time. During angular motion results will differ from those achieved with -pshiftd, but when the subject is moving in a straight line the results will be the same.

-vidres: video resolution (pixels per cm). **NOTE**: It is critical to set this variable correctly for calculation of distances and speeds, particularly when the camera zoom has been changed.

-vidxmax [pixels]: the highest possible x-position value - this is basically the dimensions of the video

signal, across.

-vidymax [pixels]: as above, for y-positions

Behavioural filtering

-set_filter [0|1]: determines if the following filter settings are applied (1) or not (0).

-f_start [time (s)]: time in seconds from the beginning of the video tracking record before which data is filtered. Neuralynx data records do not necessarily start at time "zero", so for Neuralynx data f_start is incremented by the time of the first video tracking record. For example, if f_start is set by the user to 300 (5 minutes) but the first video record is at time 1000, then f_start is reset to 1300.

-f_end [time (s)]: time in seconds from the beginning of the video tracking record after which data is filtered. See also f_start.

-f_velmin [speed,cm/s]: sets the minimum velocity for behavioural filtering of data. This variable is also used as the threshold for defining immobility, and for determining if the subject stops in the middle of a run.

-f_velmax [speed, cm/s]: as above, but this is the maximum running speed allowed.

-f_dirchoice [degrees]: sets the angle the rat must be running at for data to be included. East = 0, North = 90, West = 180, South = 270. To work properly, this setting must be used in conjunction with -f_dirtol.

f_dirtol [degrees]: maximum allowable +/- deviation from f_dirchoice allowed for inclusion. Setting this to "180" therefore guarantees that all directions are acceptable. Default is 180.

-f_avel [-1|0|1]: sets the allowable type of angular velocity (angular motion). "-1" accepts clockwise motion, "1" accepts counter-clockwise motion, and "0" accepts either. "0" is the default.

EEG Processing

-echantot [number]: (Csicsvari files only). This variable defines the total number of channels interlaced in binary EEG files. If this variable is set incorrectly, CRUNCH will not be able to read Csicsvari EEG files properly. **NOTE**: Csicsvari EEG files usually have one channel which carries a synchronization signal - this channel should be included in the total.

-echannel [number]: (Csicsvari files only). For eeg data in which multiple channels are combined in a binary file, this determines which channel is to be used. Sets the EEG channel which will be used for oscillation fitting. Other interlaced EEG channels will be ignored. In the spike output, the instantaneous phase of oscillations on this channel will be listed as an attribute of individual spikes. **NOTE**: remember that EEG channel numbers are zero-offset. That is for a 64-channel recording, the EEG channels range from 0-63.

-efreq [sampling frequency (Hz)] : (Csicsvari files only) define the EEG sampling frequency so sample counts can be converted into timestamps.

-emin [cycles/s]: The lower frequency bound of the range of oscillations CRUNCH will fit. After calculating the interval "*i*" between two successive cycles, CRUNCH will check that $1/i >$ -emin. The default is 4.

-emax [cycles/s]: The upper frequency bound of the range of oscillations CRUNCH will fit. After calculating the interval "*i*" between two successive cycles, CRUNCH will check that $1/i <$ -emax. The

default is 12.

-elowcut [cycles/s]: the cutoff for the low-cut Butterworth applied before EEG fitting - frequencies below -elowcut will be reduced. The default is 2.

-esmooth [seconds]: The size of the boxcar-averaging window used to smooth the EEG signal to remove high-frequency components of the signal. Smoothing will slightly reduce the amplitude of the oscillation. One-half the shortest interval oscillation to be fit is recommended. For example, for 4-12 Hz theta oscillations, the cycle length for 12 Hz is 0.083, so 0.04 is the default value for -esmooth.

-efit [value]: this sets the maximum mean square error for the sample oscillation compared with a frequency and amplitude-matched sine-wave - mean square errors above this value will result in rejection of the oscillation. Mean square error values can vary from zero for sampled cycles which are a perfect sine wave, to approximately 0.8 for cycles which are a poor approximation of a sinusoid. Setting -efit to 0.4 will require a very good fit but may reject many cycles. 0.6 is the default.

-halffit [0|1]: determines whether only the negative portion of the cycle is tested for goodness of fit (1), or the entire cycle (0). The default is to fit the entire cycle (0).

-escale [value]: this is a multiplication factor applied to the EEG signal before any processing is done. By setting this value to “-1”, this can be used to invert the EEG signal, which may be necessary to correct inversion of the signal if it happened during acquisition. For example, some software subtracts the EEG signal from a reference signal to remove common noise, but this will result in the data being “upside down”. The default for -escale is “1”, which does not change the EEG signal.

-estart [seconds]: If the -outeeg variable is set to “1”, CRUNCH produces a sample of post-processing EEG data to allow diagnostic analysis of things like the quality of oscillation fitting. -estart defines the number of seconds into the EEG record at which this output begins. The default is zero.

-edur [seconds]: Used in conjunction with -estart, this variable defines the number of seconds of EEG output. the default is 3.

-phaseadjust [degrees]: Because CRUNCH uses the zero-crossing on the descending portion of oscillations as phase “zero”, the phase values produced will not correspond with phase values produced by software which takes the peak or trough of the cycle as phase “zero”. -phaseadjust allows the user increment phase values by a number of degrees (0-359) to compensate. For example, setting -phaseadjust to “90” will approximately make the trough of the cycle phase zero. Note however that this can introduce an error into the phase values if the oscillation is asymmetric. The default is zero.

Spike Processing

Spike processing simply involves storing the time, probe number and cluster number for each action potential - there is no transformation of these values. An additional unique cell id is assigned to each probe/cluster combination, which begins with "1" for unclustered spikes in the Csicsvari file format, and "0" for other file formats.

-setburst defines the maximum interval (in seconds) between successive spikes for those spikes to be considered members of a burst. A typical value is 6 ms (0.006).

For the Csicsvari file format the -spikefreq variable identifies the sampling frequency for spikes - this is essential so that the original timestamps can be properly converted to seconds.

Place Map Variables

-maptype [0|1|3]: defines what type of place map will be produced (no maptype 2 at present!).
Map type “0” is the default blue-red map style, autoscaled to the peak firing rate. “1” is a yellow-purple map style which assigns a fixed proportion of pixels to each colour. This map style does not skew colours to the low end of the spectrum when spuriously high firing rates produce unusually high peak firing rates.
Map type “1” also assigns a unique colour (yellow) to bins where the cell fired no spikes, and so doubles as a spike distribution map. Map type “0” is more aesthetically pleasing in colour, while map type “1” translates better in black & white printing. **NOTE**: to use map type “1” properly, mapsmooth should be set to zero.
Map type "3" produces line-graph maps for data which is best described as one-dimensional.

-mapbinsize [cm]: the size in cm of the bins over which spikes and dwell-times are integrated to create firing rates. 2.5 cm is recommended. Smaller values create more detailed maps, at the expense of the accuracy of the actual rates generated. Reductions in mapbinsize may consequently require increases in mapsmooth.

-mapsmooth [bins]: sets the smoothing factor for place maps. Map smoothing converts the firing rate in each map bin to a new rate which is intended to better reflect the “true” firing rate in that region. The more sparse the sampling of anenvironment, or the smaller the bins used to construct the rate map, the more smoothing is required to compensate for the limited sampling in each bin. The smoothing factor is the number of bins out from the reference bin used to calculate the average. For example, setting -mapsmooth to “1” will create averages for each bin based on 3x3 bins in total (the bin plus its 8 neighbours), “2” will average 5x5 bins, and so on. -mapsmooth is automatically reset to “0” (no smoothing) for maptype 1.

-mapsmtype [1|2|3]: the type of smoothing applied to place maps. All types use a sliding 2D window to integrate cell firing from a group of adjacent map bins. Each bin is smoothed only once, and smoothed bins are not used for calculating the rates in subsequent bins.

Type 1 uses a boxcar averaging method, converting the firing rate in each bin to the mean firing rate in the block of adjacent bins centered on that bin. This method has the advantage of making false peaks in the maps apparent - a single “misplaced” high-rate bin will produce a distinct square pattern in the rate map. In contrast, Gaussian smoothing (type 3) will make this type of artefact less apparent.

Type 2 sums the spikes and dwell-times in the block of bins separately, then divides the spike total by the dwell-time total. Similar to Type 1, this method may occasionally reduce the incidence of bins with artificially high firing rates due to low dwell-times. It is conceptually the same as using a sliding window to increase the bin size used for the maps.

Type 3 uses a Gaussian kernel to smooth the rate map - it is similar to Type 1, except bins closer to the central bin are assigned greater weight in calculating the average. This method tends allows a higher degree of local smoothing without dramatically changing the overall size or shape of place fields.

-mapmindwell [samples] defines the number of position samples required in a given bin fir the rate in that bin to be included in the place map. The purpose is to avoid ridiculously high firing rates arising because a small number of spikes occurred by change in a bin the subject visited very briefly. The recommended value is 2, though this may have to be increased in linear environments, which are more prone firing rate artefacts.

-mapratepeak [-1|Hz]: for colour-scaling the rate maps, you can manually define the peak rate (spikes/s) or set the program to autoscale (-1) - that is, rates are expressed as a percentage of peak firing rate bin.. The latter is the default, though it is sometimes useful to manually set the ceiling for firing rates.

-mapthresh [%]: sets the firing rate threshold (percentage of peak firing rate) for determination of the edge of place fields. The default is 10.

-mappeakzone [%]: similar to above, but this is the threshold (percentage of peak firing rate) for inclusion of bins in the “peak zone” for a place field. The default is 75%. This value is used to define the zone the

animal must run through for it to be considered a pass through the field centre. Note however that the peakzone consists of any bins exceeding the threshold - they do not have to be contiguous.

-mapcompact [0|1]: print compact output (1) or have each probe begin on a new line (0). The latter is useful if -mapnums is also set to zero.

-mapnums [0|1]: defines the numbering scheme for place maps. “0” prints the probe and cluster number at the top of each map. “1” prints the unique cell number instead.

Output options

A description of the file structure associated with the following output options is found in Appendix A. For these options, zero = “no”, and 1 = “yes”.

- *crunch_eeg.txt*: - time range is set by -estart and -edur. crunch_eegvolts.txt
- *crunch_cycles.txt*: start and end-times for every fit cycle meeting behavioural criteria
- *crunch_eegvolts.txt*: a representation of the mean detected theta cycle

???
-outruns [0|1]: analyse runs through place fields for each cell and output *cruch_runs.txt* file characterising each run.

-outmap [0|1]: analyse place fields and produce postscript summaries.

-outfield [0|1]: substitute firing rate maps with representations of the place field for each cell. These maps show the place field in green, with the peak zone in red. Note that the pixel identified as the “peak” in the field maps will not accurately indicate the bin with the highest firing rate. Note that this function only works if -outmap is set to “1”.

-outspikes [0|1]:produce *crunch_spikes.txt* spike output. This file contains one line per spike identifying it and describing a list of parameters such as instantaneous firing rate, current position, firing phase and so on.

-outisi [0|1]: calculate interspike intervals, instantaneous firing rate and burstiness. This will not affect the insterspike interval histograms in the place field output, but will eliminate the extra calculation required for inclusion of these statistics in the *crunch_spikes.txt* file.

-outpos [0|1]: produce diagnostic position output.

-outeeg [0|1]: writes the full EEG record to an ASCII file (crunch_eeg.txt)

-outmatrix [0|1]: produce dwell/spike matrices for maps - *crunch_matrix.txt*. This includes a dwell time matrix at the top of the file and labeled firing rate matrices for each cell. The dimensions of the matrices are noted in the comment line for the dwell time matrix.

Appendix B: Output File Formats

CRUNCH produces output appropriate to the data given it. For each input file type (position, EEG, spikes), there is a corresponding output file which includes a single record for each input record. There are also files which summarise the results of all analyses, cell-specific data, and data regarding runs through place fields.

crunch_cells.txt

This file contains a trial summary of the firing characteristics of each cell (cluster) from each probe. Included are basic firing characteristics, position-related characteristics, and EEG oscillation-related characteristics. If position or EEG data were not included, the related fields will be contain a "-".

01. probe - the electrode the cell was recorded from
02. clust - the cluster number of the cell (cluster zero is omitted)
03. cellid - the unique identification number for this cell. This is a single number which can differentiate all cells in a given recording
04. burst - busrtiness of the cell
05. dwell - total trial duration in seconds, not including filtered epochs (will be the same for all cells)
06. spks - the total number of spikes fired by the cell
07. mean - mean firing rate of the cell over the entire trial
08. peak - firing rate in peak bin
09. fdwell - total time spent in place field
10. fspikes - total spikes inside place field
11. fsize - size of the place field in cm^2 = (number of bins x binsize)^2
12. runs - number of runs through place field
13. x - x-position of the place field peak
14. y - y-position of the place field peak
15. angle - angular position of place field peak
16. info - spatial information content of the firing rate map
17. spars - sparsity of the firing rate map
18. coh - spatial coherence of the firing rate map
19. phase - circular mean firing phase (range 0-359.999)
20. vect - mean length of the resultant angular vector (range 0-1)
21. p - Rayleigh test probability - is there significant clustering around the mean (i.e. a unimodal clustering)?
22. codes - text defined by the -suffix variable, for use in identifying the data lines beyond probe and cell number

crunch_eeg.txt

If -outeeg is set to "1", this file will contain one record per input EEG record, but with additional variables. For periods when the behavioural filter criteria were not met, values are replaced by "-"

01. time - time (seconds) when this EEG sample was taken
02. val - A/D value of EEG
03. smooth - smoothed version of "val"
04. phase - instantaneous theta phase
05. ifreq - instantaneous theta frequency
06. vel - velocity (if available)

crunch_eeg.plt

A gnuplot script file which plots the oscillation fitting for *crunch_eegdiag.txt* and the mean detected oscillation waveform from *crunch_eegmean.txt*.

crunch_eegdiag.txt

Similar to *crunch_eeg.txt*, but just a chunk of EEG data to illustrate oscillation fitting. The chunk is -edur seconds long, starting -estart seconds into the recording.

01. time - time (seconds) when this EEG sample was taken
02. val - A/D value of EEG
03. smooth - smoothed version of "val"
04. phase - instantaneous theta phase
05. ifreq - instantaneous theta frequency
06. vel - velocity (if available)

crunch_eegmean.txt

A representation of the mean detected theta cycle - that is, the average voltage at each phase. Means are based only on EEG data occurring when the behavioural filtering criteria are met.

01. phase
02. n
03. mean voltage

04. SEM

crunch_eegcycles.txt

Start and end times (seconds) for every theta cycle, and an indication of whether the cycle has passed the behavioural filters (if set)

01. start-time
02. end-time
03. filter - does this cycle fail the behavioural filter criteria? (0=no, 1=fail)

crunch_matrix.dat

This file contains 2-dimensional matrix representations of the dwell times (seconds) and firing rates (spikes/second) for each bin in the firing rate maps. The file will only be produced if a position file passed to CRUNCH and -outmatrix is set to "1". If there is no spike file, only the dwell time matrix will be produced. The dwell time matri will always be the first matrix in the file, and will be preceded by a comment line like this...

Dwell map: 55 bins wide

The firing rate maps will have the same dimensions as the dwell time map. The preceding example indicates that the maps are 55 x 55 bins. Each firing rate matrix wil be preceded by a comment line like this...

Probe 01 Cluster 02 Cell 2

Here "Cell" refers to the unique cell id for each probe/cluster combination.

crunch_pos.dat

If -outpos is set to "1", this file contains one line for every video tracking record.

01. time - the time (seconds) when this position was sampled
02. xpos - cartesian x-coordinate
03. ypos - cartesian y-coordinate
04. LEDdist - distance between LEDs (currently not implimented)
05. vel - instantaneous velocity (cm/s)
06. dir - movement-direction (0-369, 0=east)
07. hdir - head direction from multi-spot tracking (currently not implimented)
08. angle - angular position (0-359) relative to centre of behavioural area
09. avel - angular velocity (degrees/s)
10. filter - does this sample fail the behavioural filter criteria? (0=no, 1=fail)

crunch_runs.txt

If -outruns is set to "1", this file contains data about the runs through each cell's place field. This file will not be produced if spike or position data is not included.

01. probe - the electrode the cell was recorded from
02. clust - the cluster number of the cell (cluster zero is omitted)
03. run - the run number (zero = first run)
04. start - time at the start of the run
05. end - time at the end of the run
06. dur - duration of the run (seconds)
07. len - length of the run (cm)
08. vel - mean velocity in the run (cm/s)
09. dir - circular mean heading in run (degrees)
10. dirsd - circular standard deviation of dir - < 45 tends to indicate straight runs, > 45 tend to be convoluted runs
11. avel - mean angular velocity in the run (degrees/s)
12. spikes - number of spikes fired during the run
13. rate - the firing rate of the cell during the run (spikes/duration)
14. centre - did the run include the field peak-area? (0=no, 1=yes)
15. maxbin- what was the proportion of the peak firing rate in the map bin crossed with the highest firing rate?

16. stop - did the run include a period when velocity dropped below -f_velmin? (0=no, 1=yes)
17. code - text defined by the -suffix variable, for use in identifying the data lines beyond probe and cell number

crunch_spikes.dat

If -outspikes is set to "1", this file will contain one line for every spike fired when behavioural filtering criteria were met.

01. time - the time (seconds) at which spike occurred
02. probe - probe number
03. clust - cluster (cell) number
04. cellid - unique cell number identifying all probe/cluster combinations
05. burst - if spike is a member of a spike-burst, which spike number in the burst is it (otherwise, -1) (-out_isi must be set to 1)
06. irate - instantaneous firing rate - the number of spikes occurring within a "2 x theta" (250 ms) window centred on a spike, divided by 250 ms
07. phase - the phase of the current theta cycle at the time of the spike (0-360, -1 if theta was not detected)
08. xpos - x-position (cm) at which the spike fired
09. ypos - y-position (cm) at which the spike fired
10. angle - the angle of the spike position, relative to the centre of the environment (see -cx/-cy)
11. dir - the movement heading (0-359 degrees) at the time of the spike
12. headdir- angular orientation (0-359) of the head at the time of the spike (-1 if head direction was not detected)
13. vel - running speed (cm/s) at the spike time
14. avel - angular velocity (degrees/s) calculated based on change in angle from previous to current spike
15. run - number of the current pass through the place field
16. runtime - time elapsed since last entry to the field
17. rundist - cumulative distance travelled since last entry to place field
18. field - is the rat outside the place field (-1,0), in the field (1), or in the top firing rate portion of the field (2, see -mappeakzone)
19. peakangle - the angle separating heading from place field peak: range -180 to 180, | peakangle | < 90 if running towards peak
20. peakdis - distance from the field peak (cm)
21. ratedis - formula: d*(1-(rate/peakrate)), d= -1 if running towards peak, d= +1 if running away (value -= 0 at the peak itself)

crunch_isi_p??c??.ps

Temporary postscript graphics file depicting the inter-spike interval histogram and autocorrelogram of each cell. For the autocorrelogram, the total window width is 1000 ms and each bar represents 2.5 ms. For the interspike interval histogram, the total window width is 40 ms and each bar is 0.5 ms. If -clean is set to "1", these files will be deleted.

crunch_map_p??c??.ps

As above, only this is the firing rate map for a given cluster.

crunch_SUMMARY_??.ps

This is a graphical postscript summary of each cell, consisting of place fields, inter-spike intervals, and autocorrelograms. Intermediate postscript files containing data for each individual cell are also created - for example *crunch_isi_P03C02.ps*, *crunch_map_P03_C02.ps*, and *crunch_field_P03_C02.ps*, but these intermediate files will be deleted if -clean is set to "1" (default).

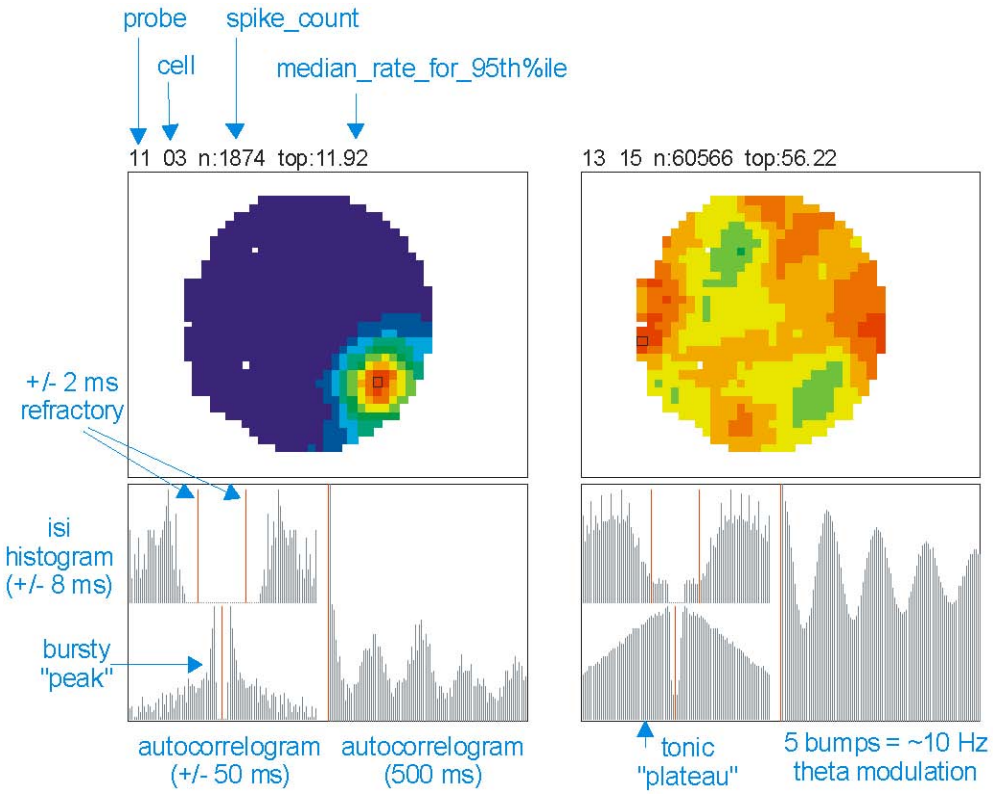
The place maps are built from *crunch_map_C??_P??.ps* files. In these maps, the black bounding box represents the camera viewing area. Coloured squares represent the firing rate of a given cell in binned positions, corrected for dwell-time at those positions. It is assumed that 256 cm is the maximum possible camera viewing dimension. If -maptype is set to "0", the map is a red-blue map autoscaled to the highest firing-rate bin (red), each colour from orange to blue representing a %10 drop from the peak rate. If -maptype is set to "1", the map is purple-yellow, with purple representing the highest firing rates. A fixed proportion of the total bins in the map are assigned to each colour, which maximises the contrast between in-field and out-of-field firing (ref???). For both map types the total spikes and peak firing rate are indicated on top of the bounding box. The peak bin itself is outlined in black.

Autocorrelation histograms are built from *crunch_isi_C??_P??.ps* files.

The large right hand histogram is a temporal autocorrelation showing the probability of spikes from a given cell recurring within a 500 ms window. Each bar covers 2.5 ms. Counting the “bumps” in the histogram yields an approximation of the rhythmic firing frequency of the cell.

The top left histogram shows the inter-spike interval (ISI) in a +/- 8 ms window. Here each bar covers 0.16 ms. Well isolated spikes from a single cell should yield a clear +- 2ms refractory period in the ISI histogram, although this may be closer to 1.5 ms for fast-spiking neurons (e.g. interneurons). The +/- 2ms refractory period is demarked with red bars. A short sharp peak on either side of the refractory period is indicative of burst-firing.

The bottom inset in the autocorrelogram is a second autocorellogram covering +/- 50 ms. At this time scale, bursty firing patterns can be seen as a sharp peak just outside of the central two bins. More tonic firing patterns appear more as a gradually declining plateau...



crunch_verbout.txt

This file contains a detailed breakdown of the progress of the analysis, including statistics regarding the position and EEG files. If setverb is set to “0”, this data is sent to a file “crunch_verbout.txt” in the program directory which is overwritten each time CRUNCH is invoked. If setverb is set to “1”, the file is placed in the data directory. If setverb is set to “2”, the output is sent to the screen.

Appendix C: useful scripts

Windows

The following script invokes CRUNCH.exe when clicked, and sets all the necessary arguments.

A function (BATCHJOB) is called, which does the bulk of the work. Add extra “CALL” lines to analyse more data.

```
@ECHO offset prog= D:\Work\Bin\Crunch\Debug\Crunch.exe
set outdir= D:\Work\Sample*\Csicsvari\crunch_cellsummary.txt

set opt1= -cx -1 -cy -1 -psmooth 0.4 -dejump 200 -emin 4 -emax 12 -esmooth 0.04 -elowcut 2
set opt2= -efit 0.6 -halffit 0 -escale 1.0 -estart 120 -edur 3 -phaseadjust 0
set opt3= -maptype 0 -mapnums 0 -mapcompact 1 -mapsmtype 3 -mapsmooth 4 -mapmindwell 2 -mapratepeak -1 -mapthresh 10 -
mappeakzone 80
set opt4= -set_filter 1 -f_velmin 5 -f_velmax 250 -f_dirchoice 360 -f_dirtol 180 -f_avel 0
set opt5= -outpos 1 -outeeg 1 -outspikes 1 -outruns 1 -outmap 1 -outisi 1 -outfield 0 -outmatrix 1
set opt6= -datatype 2 -clean 1 -setverb 2 -pause 0 -suffix "-"
set opts= %opt1% %opt2% %opt3% %opt4% %opt5% %opt6%
::************************************************************
CALL :BATCHJOB "D:\Work\Sample Data\Csicsvari" jc21-2002_23.whl jc21-2002_23.res.10 jc21-2002_23.res.11 jc21-
2002_23.res.12
PAUSE
GOTO :eof

::************************************************************
:: This function needs at least one prallel
:BATCHJOB

cd /D %1%
echo %prog% %2. %3. %4. %5. %6.
%prog% %2 %3 %4 %5 %6 %opts%
type crunch_cells.txt > ::*****************************************************************
```

Linux/Unix

Basic invocation script

It is convenient to use a script to invoke CRUNCH.exe, because in this way you can insure the full path to the program is specified. The invocation script can then be placed somewhere in the PATH for users, so that to run the program from within any directory they need only type the name of the script. This script can also pass additional arguments the user may pass on to the program.

```
#!/bin/bash
/usr/local/bin/Crunch/CRUNCH.exe $@
```

Type “echo $PATH” to see a list of directories in which this script may be placed so as to be accessible from anywhere in the file system.

Per-user clever invocation script

This script sets user-defaults and reads a file CRUNCH.set in the data directory to set session-specific overrides. The script also allows the user to enter a single base filename which, in this case, is completed with the appropriate Csicsvari file format extensions for each of EEG, spike and position data types, as well as the probe number indicator. In addition, the script accepts a word comprised of “e”, “s”, and/or “p” to indicate whether EEG, spikes, or position files (or a combination) should be read. This script actually calls the invocation script described above, which must be in the user’s PATH.

```
#!/bin/bash

# adjust the following as appropriate ################################

def1="-pause 0 -clean 1 -setverb 2 -suffix - "
def2="-vidres 1.980 -vidxmax 384 -vidymax 288 -posfreq 39.0625 -cx -1.00 -cy -1.00 -nox 0 -noy 0 -psmooth 0.400 -dejump
200.00 "
def3="-esmooth 0.040 -echannel 1 -echantot 64 -elowcut 2.00 -estart 240.00 -emin 4.00 -emax 12.00 -edur 3.00 -efit 0.60
-halffit 0 -escale 1.00 -phaseadjust 0 "
def4="-mapbinsize 2.50 -maptype 0 -mapsmooth 2 -mapmindwell 2 -mapratepeak -1.00 -mapthresh 10 -mappeakzone 75 -mapnums
1 "
def5="-set_filter 1 -f_start 0 -f_end 999999 -f_velmin 5.00 -f_velmax 999999.00 -f_dirchoice 360 -f_dirtol 180 -f_avel
0.00 "
def6="-outpos 1 -outeeg 0 -outspikes 0 -outrate 0 -outmatrix 0 -outmap 1 -outfield 0 -outruns 0 -outisi 0 "

# do not modify below this line !! ###################################

if [ ! $1 ]; then
```

```
          echo
          echo "------------------------------------------------------------"
          echo "User-specific CRUNCH script"
          echo "- sets general defaults"
          echo "- reads trial settings (CRUNCH.set) in target directory"
          echo "    - lines starting with # will be ignored "
          echo "- extra overrides can be added"
          echo "- assumes spike, EEG and position data are to be analysed"
          echo
          echo "- USEAGE: hux_crunchb1 [basename] [probe][esp][extras]"
          echo "- esp defines what to analyse: (e)eg, (s)pikes, (p)osition"
          echo "- if wildcard * is used for probes, put it in quotes"
          echo
          echo "- example: hux_crunchb1 jc21-2002_23  \"*\"  es -maptype 1"
          echo "------------------------------------------------------------"
          echo
          exit
fi

if [ $# -lt 3 ]; then echo; echo "Error: program needs [basename] [probe] [esp]"; echo; exit; fi


basename=$1; probe=$2; analysis=$3; shift; shift; shift; extras=$@
prog=hux_crunch

let count=0
if [ $(echo "$analysis" | grep "e" -c) != 0 ]; then let count=1; infiles=$infiles" "$basename.eeg; fi
if [ $(echo "$analysis" | grep "s" -c) != 0 ]; then let count=1; infiles=$infiles" "$basename.res.$probe; fi
if [ $(echo "$analysis" | grep "p" -c) != 0 ]; then let count=1; infiles=$infiles" "$basename.whl; fi
if [ $count -le 0 ]; then echo; echo "Error: argument \""$analysis"\" specifies neither e,s, nor p"; echo; exit; fi

arguments=$def1""$def2""$def3""$def4""$def5""$def6
while read temp
do
          if [ $(echo "$temp" | grep "#" -c) == 0 ] ; then
                    arguments=$arguments" "$temp
          fi
done < CRUNCH.set
arguments=$arguments" "$extras
$prog  $infiles  $arguments
```